Санкт-Петербургский Политехнический Университет Петра Великого  
Институт Компьютерных наук и технологий  
Кафедра компьютерных систем и программных технологий

Лабораторная работа 7

Предмет: Проектирование реконфигурируемых гибридных вычислительных систем

Тема: Pipelining for Performance Demo Script

Задание 1

Студент: Безрукова Ю.С.

Гр. № 3540901/81502

Преподаватель: Антонов А.П.

Санкт-Петербург

2019

Санкт-Петербургский Политехнический Университет Петра Великого  
Институт Компьютерных наук и технологий  
Кафедра компьютерных систем и программных технологий

Лабораторная работа 7

Предмет: Проектирование реконфигурируемых гибридных вычислительных систем

Тема: Pipelining for Performance Demo Script

Задание 1

Студент: Белоглазов К.И.

Гр. 3540901/81501

Преподаватель: Антонов А.П.

Санкт-Петербург

2019

Оглавление

[Ход работы 4](#_Toc24820336)

[Решение 1 4](#_Toc24820337)

[Решение 2 9](#_Toc24820338)

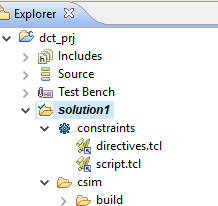
[Решение 3 12](#_Toc24820339)

[Вывод 14](#_Toc24820340)

# Ход работы

## Решение 1

Открытие проекта



Файл с исходным кодом

**#include** "dct.h"

**void** **dct\_1d**(dct\_data\_t src[DCT\_SIZE], dct\_data\_t dst[DCT\_SIZE])

{

**unsigned** **int** k, n;

**int** tmp;

**const** dct\_data\_t dct\_coeff\_table[DCT\_SIZE][DCT\_SIZE] = {

**#include** "dct\_coeff\_table.txt"

};

DCT\_Outer\_Loop:

**for** (k = 0; k < DCT\_SIZE; k++) {

DCT\_Inner\_Loop:

**for**(n = 0, tmp = 0; n < DCT\_SIZE; n++) {

**int** coeff = (**int**)dct\_coeff\_table[k][n];

tmp += src[n] \* coeff;

}

dst[k] = DESCALE(tmp, CONST\_BITS);

}

}

**void** **dct\_2d**(dct\_data\_t in\_block[DCT\_SIZE][DCT\_SIZE],

dct\_data\_t out\_block[DCT\_SIZE][DCT\_SIZE])

{

dct\_data\_t row\_outbuf[DCT\_SIZE][DCT\_SIZE];

dct\_data\_t col\_outbuf[DCT\_SIZE][DCT\_SIZE], col\_inbuf[DCT\_SIZE][DCT\_SIZE];

**unsigned** i, j;

// DCT rows

Row\_DCT\_Loop:

**for**(i = 0; i < DCT\_SIZE; i++) {

dct\_1d(in\_block[i], row\_outbuf[i]);

}

// Transpose data in order to re-use 1D DCT code

Xpose\_Row\_Outer\_Loop:

**for** (j = 0; j < DCT\_SIZE; j++)

Xpose\_Row\_Inner\_Loop:

**for**(i = 0; i < DCT\_SIZE; i++)

col\_inbuf[j][i] = row\_outbuf[i][j];

// DCT columns

Col\_DCT\_Loop:

**for** (i = 0; i < DCT\_SIZE; i++) {

dct\_1d(col\_inbuf[i], col\_outbuf[i]);

}

// Transpose data back into natural order

Xpose\_Col\_Outer\_Loop:

**for** (j = 0; j < DCT\_SIZE; j++)

Xpose\_Col\_Inner\_Loop:

**for**(i = 0; i < DCT\_SIZE; i++)

out\_block[j][i] = col\_outbuf[i][j];

}

**void** **read\_data**(**short** input[N], **short** buf[DCT\_SIZE][DCT\_SIZE])

{

**int** r, c;

RD\_Loop\_Row:

**for** (r = 0; r < DCT\_SIZE; r++) {

RD\_Loop\_Col:

**for** (c = 0; c < DCT\_SIZE; c++)

buf[r][c] = input[r \* DCT\_SIZE + c];

}

}

**void** **write\_data**(**short** buf[DCT\_SIZE][DCT\_SIZE], **short** output[N])

{

**int** r, c;

WR\_Loop\_Row:

**for** (r = 0; r < DCT\_SIZE; r++) {

WR\_Loop\_Col:

**for** (c = 0; c < DCT\_SIZE; c++)

output[r \* DCT\_SIZE + c] = buf[r][c];

}

}

**void** **dct**(**short** input[N], **short** output[N])

{

**short** buf\_2d\_in[DCT\_SIZE][DCT\_SIZE];

**short** buf\_2d\_out[DCT\_SIZE][DCT\_SIZE];

// Read input data. Fill the internal buffer.

read\_data(input, buf\_2d\_in);

dct\_2d(buf\_2d\_in, buf\_2d\_out);

// Write out the results.

write\_data(buf\_2d\_out, output);

}

Файл- тест

// Copyright (C) 2008 AutoESL Design Techonologies, Inc.

// All rights reserved.

**#include** <stdio.h>

**#include** "dct.h"

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**int** **main**() {

**short** a[N], b[N], b\_expected[N];

**int** retval = 0, i;

FILE \*fp;

fp=**fopen**("in.dat","r");

**for** (i=0; i<N; i++){

**int** tmp;

**fscanf**(fp, "%d", &tmp);

a[i] = tmp;

}

**fclose**(fp);

fp=**fopen**("out.golden.dat","r");

**for** (i=0; i<N; i++){

**int** tmp;

**fscanf**(fp, "%d", &tmp);

b\_expected[i] = tmp;

}

**fclose**(fp);

**dct**(a, b);

**for** (i = 0; i < N; ++i) {

**if**(b[i] != b\_expected[i]){

**printf**("Incorrect output on sample %d. Expected %d, Received %d \n", i, b\_expected[i], b[i]);

retval = 2;

}

}

**#if** 0 // Optionally write out computed values

fp=fopen("out.dat","w");

**for** (i=0; i<N; i++){

fprintf(fp, "%d\n", b[i]);

}

fclose(fp);

**#endif**

**if**(retval != (2)){

**printf**(" \*\*\* \*\*\* \*\*\* \*\*\* \n");

**printf**(" Results are good \n");

**printf**(" \*\*\* \*\*\* \*\*\* \*\*\* \n");

} **else** {

**printf**(" \*\*\* \*\*\* \*\*\* \*\*\* \n");

**printf**(" BAD!! %d \n", retval);

**printf**(" \*\*\* \*\*\* \*\*\* \*\*\* \n");

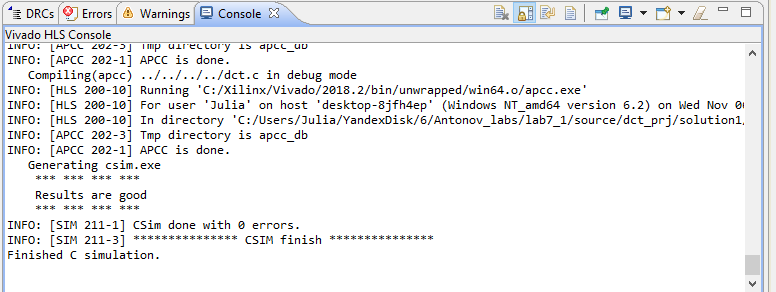
}

**return** retval;

}

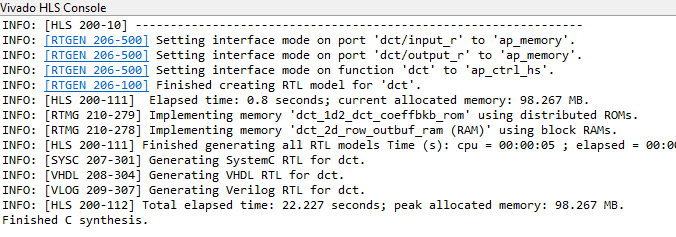
Данный исходный файл использует дискретное косинусное преобразование (DCT). Функция реализует алгоритм 2D DCT, сначала обрабатывая каждую строку входного массива через 1D DCT, затем обрабатывая столбцы результирующего массива через тот же 1D DCT. Он вызывает функции read\_data, dct\_2d и write\_data. Функция read\_data определена в строке 54 и состоит из двух циклов: RD\_Loop\_Row и RD\_Loop\_Col. Функция write\_data определена в строке 66 и состоит из двух циклов для выполнения записи результата. Функция dct\_2d, определенная в строке 23, вызывает функцию dct\_1d и выполняет транспонирование. Наконец, функция dct\_1d, определенная в строке 4, использует dct\_coeff\_table и выполняет требуемую функцию, реализуя базовую итеративную форму алгоритма DCD 1D типа II.

Моделирование

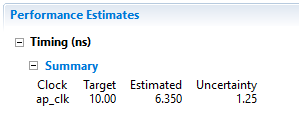
\

Результат моделирования успешный.

Синтез

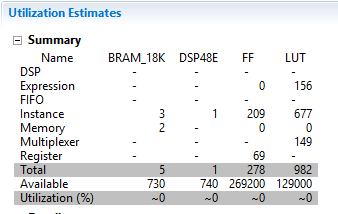


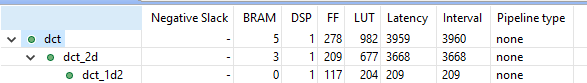
Производительность

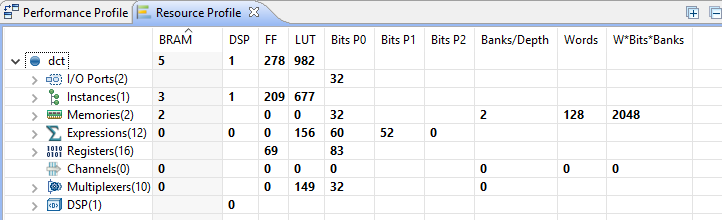


Полученная величина задержки укладывается в заданное значение.

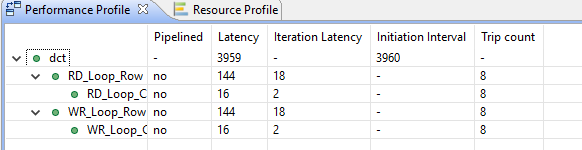
Использование ресурсов



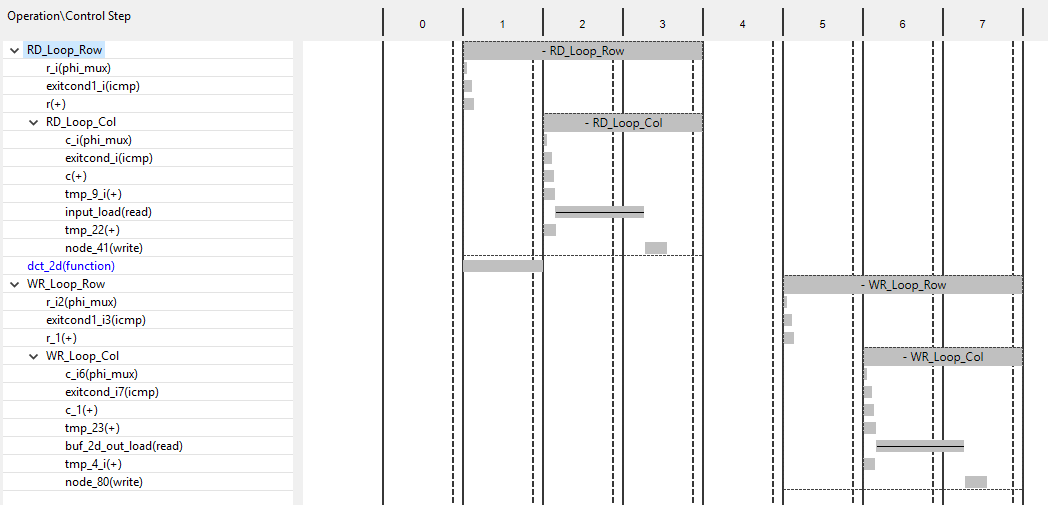




Данное решение потребует на микросхеме 982 элемента LUT, 278 регистров, 1 DSP48E и 5 блоков RAM\_18K.



Задержка составляет 3959 тактов, интервал инициализации – 3960.

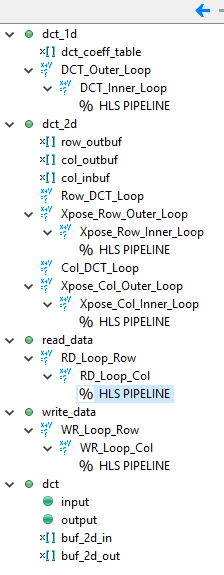


Максимальная величина задержки для dct\_2d равна 3668. Максимальная величина задержки для dct\_1d2 равна 209.

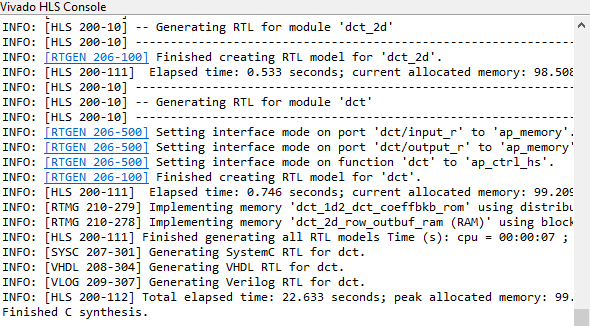
## Решение 2

Создание решения 2

Применение директивы Pipeline.

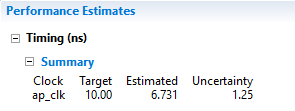


Синтез



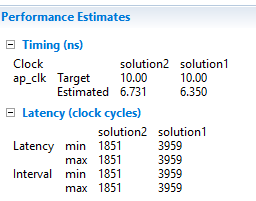
Отчет

Производительность



Величина задержки укладывается в заданное значение.

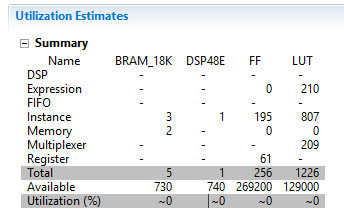
Сравнение решений



На изображении выше видно значительное уменьшение количества тактов в задержке получения решения и интервале инициализации.

Максимальное значение задержки для второго решения составляет 1851 такт.

Использование ресурсов

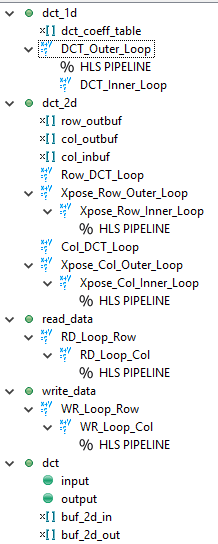


Данное решение потребует на микросхеме 1226 элементов LUT, 256 регистров и как и в первом решении 1 DSP48E и 5 блоков RAM\_18K.

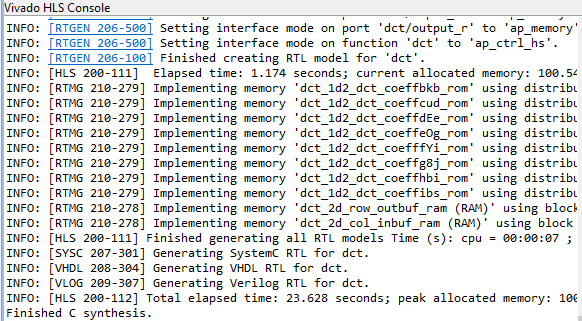
Максимальное значение задержки для dct\_2d равно 1718. Максимальное значение задержки для dct\_1d2 равно 97.

## Решение 3

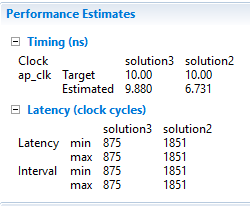
Изменение директив

****

Синтез



Сравнение решений



На изображении выше видно, что время задержки третьего решения значительно выше, но при этом, количество тактов практически в 2,5 раза меньше.

# Вывод

В ходе выполнения работы было получено три решения и проведено сравнение величин задержек в этих решениях. Директива PIPELINE встраивает конвейерные регистры в сгенерированный RTL, чтобы улучшить значение задержки. При правильном применении этой директивы есть возможность увеличить пропускную способность проекта.